Applicant's claim 22 recites:

> 22. A method of resolving condensed computer code having a plurality of types of code structures, each of the types of code structures including a plurality of index references, the method comprising the steps of:
>
> reading a list of identifiers for each type of code structure, each list including an index reference corresponding to each of the identifiers in the list; and
>
> replacing each of the index references in the computer code with the respective identifier corresponding to each respective index reference.

Hamby, on the other hand, teaches a method for building (or compiling) a software program that has "no unresolved external references". For example, Hamby states in column 6, lines 18-24:

> . . .After creating the source code files which define a given executable file (e.g., the entire program) or a self-contained portion of ***a program having no unresolved external references*** (e.g., a dynamic link library), the developer invokes the debugger to debug the source files. The debugger then programmatically invokes the Incremental Builder of the present invention. . .
>
> (Emphasis added)

While applicant's claim 22 is directed to a method for resolving "condensed computer code", which code can be received over a network more quickly due to its condensed form, Hamby's teachings are directed to a method for increasing the "computational efficiency" of code execution. Hamby never mentions the existence of "condensed computer code". Hamby does, however, make statements like the following:

> Prior byte code compilation schemes, which do not have the advantage of a persistent symbol table in which to cache the results of byte code compilation must of necessity compile all of the byte code which is ultimately compiled every time the program is executed. The persistent symbol table taught herein enables the storage of the results of byte code compilation as code objects, thus eliminating the need to compile the code each time the program is loaded. This of course gives rise to significant advantages in computational efficiency over prior byte code compilers.
>
> Hamby, col. 9, lines 41-50.

Hamby further states:

> A code object is the ***fully translated machine language implementation of a function definition***, or the initial value of a variable, which refers to IL symbols representing other code objects by means of direct memory reference as opposed to Ids. ***Code objects and IL symbols are low level data structures*** used to create an image, i.e., the "program".

Hamby, col. 6, lines 5-10 (emphasis added).

Hamby also states:

> Where applications are distributed over a network, some new and distinct problems eventuate: the first such problem is that the size of the program must be small, so that the cost of transferring the program from the point where it resides (the server) to the point where it is to be executed (the client) is minimal. This fact typically precludes the use of normal binary images in target machine code format since these images are normally wasteful expressions of the program.

Hamby, col. 4, lines 17-24.

Finally, in stating a goal of Hamby's teachings, Hamby states:

> . . .Ideally a client side compiler would only be invoked at the point where a new program element is loaded. At all other times, the client side compiler should enable execution of the program at the same speed as a statically compiled program.

Hamby, col. 5, lines 18-22.

None of the above excerpts imply to applicant's representative that Hamby contemplates the existence of a "condensed" computer code. Although Hamby never states that his methods produce a "larger" code, it seems that Hamby is attempting to balance the *disadvantages* of machine code (e.g., larger code sizes) against the *advantages* of machine code (e.g., faster execution times).

In contrast to Hamby, applicant's claim 22 requires the existence of a "condensed computer code". The condensed computer code is then resolved by replacing "index references" (e.g., integers) found in the code with "identifiers" (e.g., classes, methods and fields).

The Examiner should note that the teachings of applicant and Hamby are related, yet distinct. Applicant's claim 22 is directed to a method for "resolving" condensed computer code. As taught in applicant's specification, condensed computer code is advantageous in that it may be transmitted and received over a network more quickly. However, once transmitted, it may be necessary to "resolve" the condensed code. Once resolved, Hamby's teachings could be applied to the code so that it could be executed more efficiently (i.e., without recompiling, or with only minimal recompiling).

Given that Hamby's and applicant's teachings are directed to different problems that present at different stages of a code's handling, and Hamby provides no suggestion that his teachings can be applied to solve the problems disclosed by applicant, it is believed that applicant's claim 22 is novel and unobvious over the teachings of Hamby.

Claim 23 is believed to be allowable at least for the reason that it depends from an allowable claim 22.

2. Restriction Requirement

The Examiner's restriction requirement is respectfully traversed. Although claims 24-27 relate to "transmitting computer code", claims 28-31 relate to "receiving and executing computer code", including "resolving" fetched operands. As a result, it is believed that claims 28-31 are sufficiently related to claims 22 and 23 such that no further search would be required to examine these claims. Applicant therefore requests that these claims not be restricted from the current application.

3. Timeliness of Response

Although this Response is being filed on November 4, 2002, this Response is believed to be filed within two months of the September 3, 2002 mailing date of the Examiner's Final Office Action (since November 3, 2002 fell on a Sunday).

CONCLUSION

Applicant requests that a timely Notice of Allowance be issued in this case.

Respectfully submitted,
DAHL & OSTERLOTH, L.L.P.

By: _____

Gregory W. Osterloth
Reg. No. 36,232
555 Seventeenth Street, Suite 3405
Denver, CO 80202
Tel: (303) 291-3200
FAX: (303) 291-3201